TFG: Detección semi-automática de nuevos pares de estrellas con movimiento propio común

Trabajo de Fin de Grado

David Antuña Rodríguez
Daniel Gutiérrez Gertrúdix
Javier Carrión García

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2017/2018

Documento maquetado con TEXIS v.1.0+.

TFG: Detección semi-automática de nuevos pares de estrellas con movimiento propio común

Trabajo de Fin de Grado

Dirigida por el Doctor
Rafael Caballero Roldán

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2017/2018

Agradecimientos

Queremos dar las gracias a nuestro director, Rafael Caballero, por el esfuerzo y la dedicación a este trabajo, y por habernos aguantado durante todo el año. Ha sido un placer haber tenido la oportunidad de trabajar no solo con un gran científico sino con una persona maravillosa.

Resumen

La detección de sistemas binarios tiene gran importancia en la astrofísica, pues permite, por ejemplo, calcular con precisión la distancia a la que se encuentran sus componentes o descubrir nubes de materia oscura mediante la detección de cambios inesperados en las órbitas. Sin embargo, la mera ocurrencia de dos estrellas cercanas en el cielo no supone que se trate de un sistema binario, porque pueden verse así por efecto de la perspectiva. Para saber con seguridad que en efecto se trata de una pareja de estrellas con un vínclulo gravitacional habría que observar cómo giran una alrededor de la otra, pero esto a menudo no es posible al tratarse de órbitas que requieren decenas de miles de años. Un mecanismo indirecto para detectar nuevas estrellas binarias es descubrir dobles con un notable movimiento propio común. La idea es que es estadísticamente improbable encontrar parejas que se muevan muy rápido, y en la misma dirección y sentido.

Este trabajo pretende colaborar en la detección de estos pares con alto movimiento propio común como posibles "candidatas" a estrella binarias. Para ello utilizamos la superposición de imágenes tomadas por telescopios profesionales con una diferencia temporal cercana a los 50 años. La detección se realiza comprobando parejas cuya posición varía de forma significativa entre ambas imágenes. Para ello hemos realizado un programa que analiza de forma automática estas imágenes y sugiere posibles candidatas que deben ser corroboradas por el usuario. El sistema descarta el 97 % de las imágenes que no tienen estrellas dobles, y encuentra el 33 % de las imágenes que sí la tienen.

Índice

Índice de figuras

Capítulo 1

Introducción

Resumen: En primer lugar vamos a aclarar qué es una estrella doble y cuál es su utilidad para la comunidad científica. También plantearemos el problema que este trabajo pretende resolver.

1.1. Estrellas dobles

A lo largo de la historia los astrónomos se han ocupado de las características de los diferentes objetos que se encuentran en el espacio: estrellas, planetas, galaxias, etc. Entre estos objetos, desde antiguo han llamado la atención las estrellas dobles.

Un pionero en este campo fue **Willen J. Luyten** quien se dedico al estudio de estrellas con movimiento propio, y descubrió gran cantidad de estrellas múltiples sentando así las bases para el estudio de las mismas. (Luyten, 1971)

Según `http://www.astromia.com/glosario/binaria.html`

> "Una Estrella doble o Estrella binaria, es una pareja de estrellas que se mantienen unidas por gravitación y giran en torno a su centro de masas común. Los periodos orbitales, que van desde minutos en el caso de dobles muy cercanas hasta miles de años en el caso de parejas distantes, dependen de la separación entre las estrellas y de sus respectivas masas. La observación de las órbitas de estrellas dobles es el único método directo que tienen los astrónomos para pesar las estrellas. En el caso de parejas muy próximas, su atracción gravitatoria puede distorsionar la forma de las estrellas, y es posible que fluya gas de una estrella a otra en un proceso denominado transferencia de masa."

Aunque la mayoría de las estrellas que vemos son dobles o incluso múltiples, muy pocas de ellas son detectables a través de un telescopio debido a la gran cercanía de sus componentes. A estas se les llama dobles visuales, y es a las que nos dedicamos en este trabajo. (Greaves, 2004)

1.2. Problema

En este trabajo nos encargaremos del marco de las estrellas con un movimiento propio común elevado, es decir, parejas de estrellas que tengan un movimiento muy rápido, en el cual las componentes, dirección y sentido, coincidan.

La localización de candidatas a estrellas binarias mediantes este procedimiento es un trabajo muy delicado por lo que se realiza de forma manual, es necesario analizar las fotografías una a una hasta que se encuentra un par de estrellas con unas características similares que encajen en el arquetipo de un sistema de estrellas dobles.

1.3. Importancia

Esta tipología de estrellas alberga una importancia clave a la hora de recolectar datos tales como el peso de una estrella, como se menciona en la sección 1.1, o en caso de ya conocer las masas su distancia exacta.

Otra ventaja de conocer estas estrellas radica en el descubrimiento de nubes de materia oscura, dado que estos sistemas de estrellas se mueven en paralelo es posible observar cambios una de las estrellas que componen el sistemas que indiquen la presencia de cuerpos invisibles a simple vista como dichas nubes. Hartkopf, Harmanec y Guinan (2007)

1.4. Estado del arte

A pesar de que, como ya hemos mencionado, estos sistemas albergan datos muy útiles el proceso de detección empleado no ha evolucionado mucho desde Luyten, en aquel entonces se superponían placas fotografías de un mismo sector espacial y se determinaban las propiedades de los distintos cuerpos celestes para comprobar si se trataba de estrellas binarias. A día de hoy, se emplean fotografías de mayor calidad, debido a los avances tecnológicos en los telescopios empleados para tomarlas, pero sigue tratándose de un trabajo manual que toma gran cantidad de horas incluso para procesar sectores minúsculos comparados con la inmensidad del espacio. (Bos, 1923)

1.5. Nuestra propuesta

Queremos crear un sistema de detección de estrellas binarias con un alto movimiento propio común y, por tanto, observables en un periodo relativamente corto de tiempo, 50 años aproximadamente. Este proceso no sería completamente automático sino que una vez detectado una posible estrella doble tendría que ser revisada manualmente para cerciorarse de que no se trata de un falso positivo.

Además, pretendemos que el filtrado de imágenes se desarrolle en múltiples pasos y diseñar un mecanismo de control de flujo que permita cambiar dichos pasos de manera sencilla.

Durante el desarrollo del proyecto hemos empleado un repositorio de la plataforma GitHub para gestionar nuestro código, se puede acceder en `https://github.com/Godh3x/tfg-doublestars`.

Capítulo 2

Introduction

Abstract: First, we present the concept of a double star and the relevance it has for the scientific community. Next, we state the problem this work intends to solve.

2.1. Double stars

Throughout history astronomers have dealt with the features of different objects in space: stars, planets, galaxies, etc. Among them, attention has been drawn to double stars since ancient times.

A pioneer in this field was **Willen J. Luyten** who dedicated himself to the study of stars with common proper motion, and discovered a large number of multiple stars, laying the foundations for future investigation. (Luyten, 1971)

According to `http://www.astromia.com/glosario/binaria.html`.

> "A Double Star or Binary Star is a pair of stars held together by gravity spinning around the common center of masses. The orbital periods, that oscillate between minutes for very close doubles to millions of years for the distant ones, depend on the separation between stars and their masses. The observation of the orbits of double stars is the only direct method the astronomers have to weight stars. For near pairs the gravitational pull may distort the shape of the stars, and gas may flow from one star to another in a process called mass transfer."

Even though most of the stars we see are double or even multiple, very few of them are measurable using a telescope due to the closeness of its components. These are called visual doubles, and that is what we are dedicated to in this work. (Greaves, 2004)

2.2. Issue

This work focuses in the frame of double stars with high proper motion, that is, pairs of stars with very fast movement in which the components, sense and direction, coincide.

The detection of binary stars is a very delicate work because it is done manually, the pictures have to be analyzed one by one looking for a pair of stars with similar characteristics that fit the archetype of a double star system.

2.3. Importance

This kind of stars have a key value to collect data such as the weight of a star, as stated in the section 2.1, or, knowing the masses, the exact distance.

Another benefit of these stars lies in the discovery of dark matter clouds, these star systems movement is parallel so it is possible to notice changes in the expected behaviour suggesting the presence of invisible masses such as said clouds. Hartkopf, Harmanec y Guinan (2007)

2.4. State of the art

As previously stated, even though these kind of systems hold very useful data the current detection process has not evolved much since Luyten, back then astronomers used to overlap photographic plates from the same space sector and study the properties of the different celestial bodies in order to check if they are or not binary systems. Nowadays the quality of the pictures have improved due to technological advances in the used telescopes, but it is still a manual job that takes a lot of hours to even process the tiniest sectors compared to the vastness of space. (Bos, 1923)

2.5. Our approach

We want to create a system capable of detecting binary stars with high proper motion and, therefore, observable in a relatively short period of time, approximately 50 years. This process wouldn't be completely automatic, once a possible double star is detected it would have to be manually checked to make sure it is not a false positive.

In addition, we intend the filtering of images to be developed in multiple steps and design a flow control mechanism that allows to change these steps in a simple way.

During the development of the project we have used a GitHub repository to manage our code, it is available in `https://github.com/Godh3x/tfg-doublestars`.

Capítulo 3

Estrellas dobles

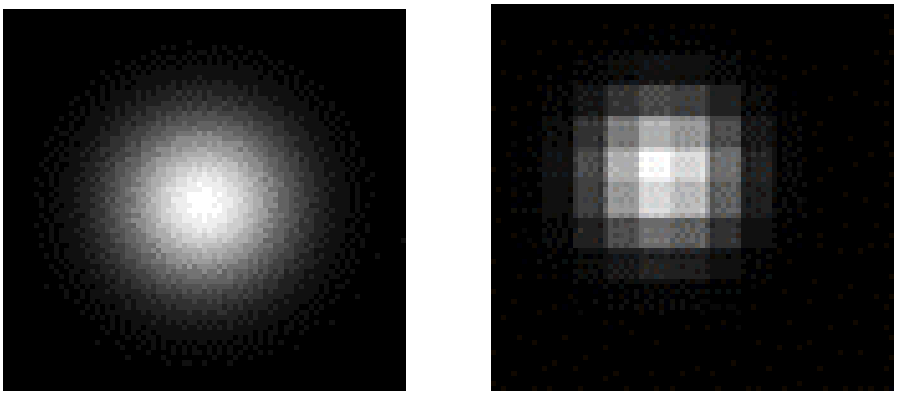
Resumen: Antes de abordar el problema que nos ocupa en este trabajo vamos a explicar los conceptos relativos a estrellas dobles que en el se tratan.

3.1. Centroides

Un punto fundamental para la detección de estrellas dobles consiste en la búsqueda y análisis de los centros de las estrellas, los llamados centroides. Aunque las imágenes de la estrella son puntuales debido a diversos factores como la atmósfera, las imperfecciones de las ópticas o las limitaciones de las cámaras, las estrellas terminan siendo manchas que ni siquiera aparecen como círculos, cuyo centro puede determinarse sin más (figura 3.1a), sino como una acumulación de puntos brillantes, con un contorno difuso (figura 3.1b). Esta problemática es incluso más notable en nuestro caso puesto que trabajamos con la superposición de dos imágenes, lo cual dificulta aún más la detección de los limites de una estrella.

3.2. Coordenadas

Un sistema de coordenadas celestes no es más que un modo de especificar la posición de un cuerpo en el espacio. A pesar de estar a diferentes distancias, éstas son tan grandes que desde la Tierra vemos las estrellas como puntos situados en la bóveda celeste, gracias a este efecto es posible el uso de, tan solo, dos coordenadas para definir la localización exacta. Existen varios sistemas de este tipo pero en este trabajo nos quedaremos con el sistema de coordenadas ecuatorial, que es el usado por los astrónomos profesionales debido a su facilidad de cálculo y precisión.



(a) (b)

Figura 3.1: Estrella ideal vs. Estrella real

Antes de explicar que compone este sistema vamos a introducir dos conceptos. En primer lugar el circulo horario de un objeto es aquel que pasa por ambos polos e intersecta al objeto. En segundo lugar el equinoccio de primavera se da en Marzo, exactamente en el momento en que los rayos solares golpean el ecuador perpendicularmente. Este equinoccio es el momento cero del tiempo sideral, usado por los astrónomos para localizar objetos en el espacio, y por tanto el momento cero de el sistema ecuatorial.

En este sistema una coordenada tiene dos componentes: la ascensión recta, RA, y la declinación, Dec. La ascensión recta, figura 3.2, mide la distancia angular, calculada hacia el este sobre la proyección del ecuador terrestre, entre el equinoccio de primavera y el circulo horario del objeto. La declinación mide la distancia angular de un objeto perpendicularmente respecto de la proyección del ecuador, positiva hacia el norte. Por ejemplo la proyección del polo norte tiene una declinación de $+90^o$ y la del sur de -90^o.

Para generar grandes cantidades de coordenadas ecuatoriales hemos creado dos generadores de coordenadas, uno continuo y otro aleatorio. Ambos generadores necesitan que se especifiquen los rangos en los que trabajar, que oscilan entre los siguientes valores, ambos en grados.

- RA $\in$ [0, 360].
- Dec $\in$ [-90, 90].

Además de estos rangos el generador aleatorio necesita el número de coordenadas deseadas, se asegurará de que no existan repetidas, en cambio el continuo necesita dos valores uno para el desplazamiento de la ascensión recta y otro para la declinación. Teniendo en cuenta que posteriormente

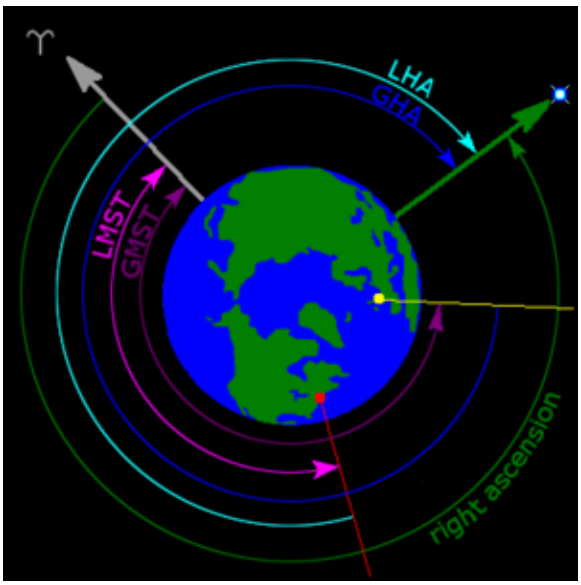


Figura 3.2: Vista cenital de la tierra, sobre el polo norte. ♈ representa el equinocio de primavera.
Fuente: `https://en.wikipedia.org/wiki/File:Hour_angle_still1.png`

usaremos un zoom x4 los valores que permiten solapar las fotos sin perder información son los siguientes.

- Desplazamiento RA: 0.02
- Desplazamiento Dec: 0.01

Al usar el generador continuo se puede ver que la diferencia en la ascensión recta no se corresponde al desplazamiento especificado, esto se debe a que estamos trabajando en un sistema esférico por lo cual debemos ajustar el desplazamiento en base a la declinación.

3.3. Atributos

Un sistema de estrellas binarias viene definido por una serie de parámetros, ilustrados en la figura 3.3, que veremos a continuación. En el sistema una de las componentes es principal y la otra secundaria, el criterio para definir cuál es cuál es simple, la más brillante es la principal.

La separación o distancia angular entre dos objetos, medida desde un punto ajeno a ambos, es el ángulo formado por las dos trazas con origen en el punto ajeno desde el que se observa y ambos objetos.

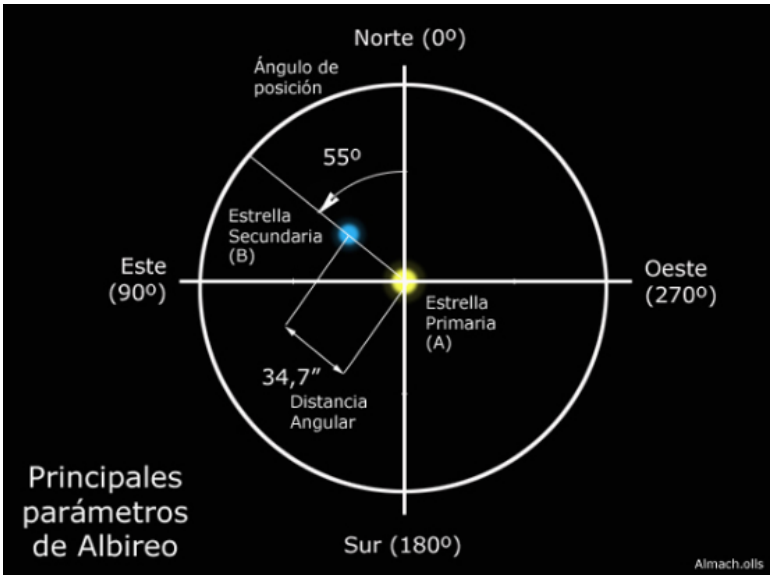


Figura 3.3: Ilustración de los parámetros de un sistema binario.
Fuente: `http://www.asociacionhubble.org/portal/modules/grupos/estrellasdobles/guiaobs/guiaobs.pdf`

El ángulo de posición, abreviado PA, se define como el desplazamiento angular de la componente secundaria, medido desde el polo norte celeste, respecto de la primaria.

La magnitud o magnitud aparente denota el brillo de una estrella observado desde la Tierra, cuanto más bajo es este valor más brillante es la estrella. Como se comentaba al principio de la sección es este parámetro el que determina que componente es principal.

Y, por último, su espectro no es más que el color que emite una estrella, este valor está directamente relacionado con la temperatura.

Capítulo 4

Recolección de datos

Resumen: A lo largo de este capítulo veremos que herramientas hemos empleado para obtener los datos astronómicos con los que trabajar, así como los de estrellas dobles ya catalogadas.

4.1. Aladin

Aladin Sky Atlas, `http://aladin.u-strasbg.fr/`, es un programa interactivo desarrollado por el **CDS**, Centro de Datos Astronómicos de Estrasburgo, que permite acceder recursos astronómicos digitalizados.

Esta herramienta no solo permite visualizar imágenes de cuerpos celestes a lo largo de los años, también permite realizar una superposición de las mismas para facilitar su comparación. Este proceso se realiza, en general, sin complicaciones pero puede que las fotografías se hallan tomado con una leve desviación de ángulo o simplemente que no determine correctamente el encuadre, como resultado surgen imágenes en las que parece que todas las estrellas se han desplazado.

En la figura 4.1 se ilustra la obtención de dos imágenes mediante este programa, en ella se pueden observar multiples zonas de interes.

(a) Permite elegir el servidor de imágenes que se quiere consultar, en nuestro empleamos *Aladin images*.

(b) Indica el objetivo de la busqueda, es decir, la coordenada de la cuál se desean obtener los recursos, una vez introducida es necesario pulsar *Submit*.

(c) En este caso podemos ver que para dicha coordenada existen múltiples imágenes de diferentes fuentes y los datos asociados a cada una de ellas,

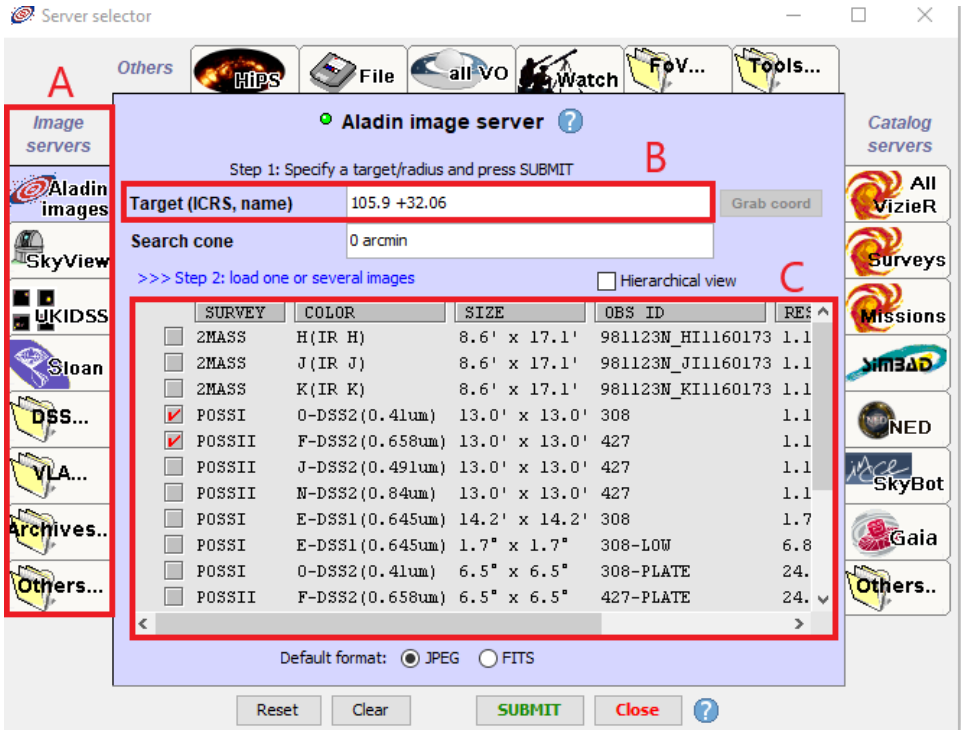


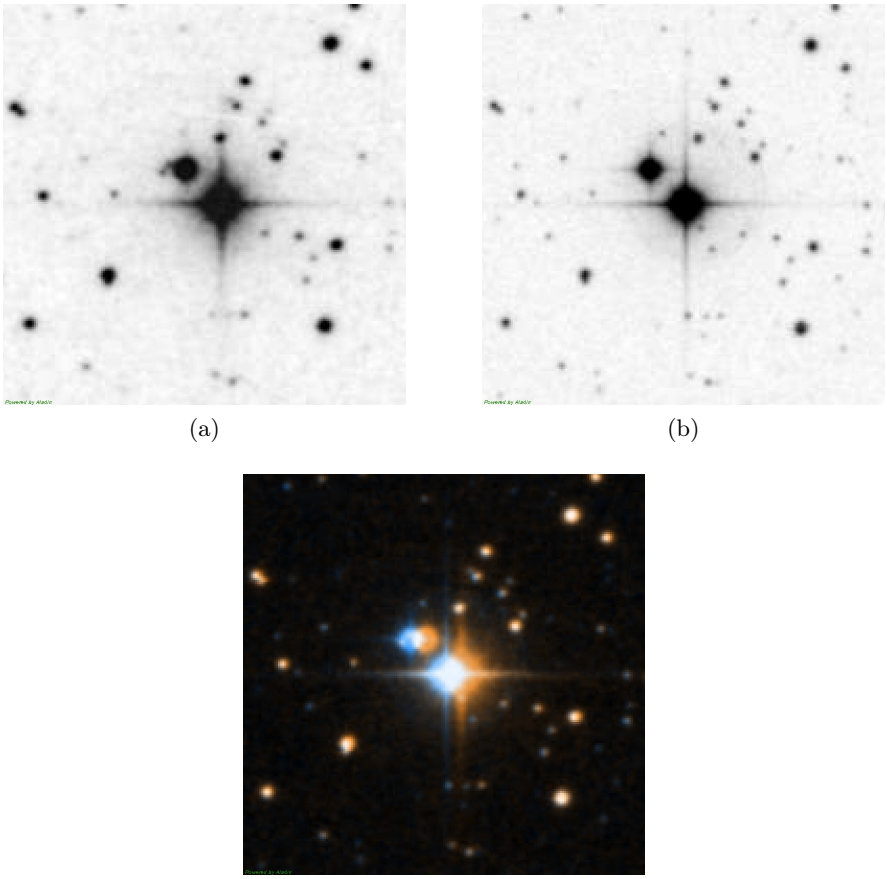
Figura 4.1: Obtencion de recursos mediante el uso de Aladin.

entre estos datos se encuentran las fechas en las que se tomaron. Para solicitarlas tan solo es necesario seleccionarlas, empleando los recuadros de la izquierda, y presionar *Submit*.

4.2. Obtención de imágenes

Como se describe la sección 4.1 es posible obtener imágenes de varias fuentes, que muestren la evolución de los cuerpos celestes en coordenadas concretas con años de diferencia.

Una vez hemos obtenido las fotografías deseadas utilizamos la herramienta RGB de Aladin que reemplaza el blanco por rojo en una imagen y por azul en la otra, que corresponde a la misma zona del cielo pero 50 después. A continuación superpone ambas imágenes, creando una nueva. Las estrellas que están en la misma posición en ambas imágenes aparecen aproximadamente blancas por la superposición de rojo y azul. En cambio, en las que ha habido un movimiento notable aparecerá una zona roja y otra azul. Si el movimiento es mayor al diámetro de la estrella, está aparecerá en la imagen compuesta como dos estrellas diferentes, una roja y otra azul. Como se puede



(a) (b)

(c)

Figura 4.2: Superposición de imágenes

apreciar, figura 4.2c, Aladin no solo superpone estás imágenes, también las rota para tratar de encajarlas.

Nuestro reto será encontrar qué "estrella roja" asociar con una "estrella azul" de forma que seamos capaces de detectar que son, en efecto, dos imágenes de la misma estrella.

Para facilitar el proceso de descarga Aladin permite emplear scripts en combinación con ficheros de parámetros. Puesto que no existe ninguna fuente que tenga fotos de todas las coordenadas celestes son necesarios dos scripts con el siguiente formato.

```
reset
reticle off
overlay off
get aladin(POSSI,O) $1
get aladin(POSSII,F) $1
```

```
sync
RGB 1 2
sync
zoom 4
sync
save /.../images/$1.jpg
```

El script realizará los siguientes pasos:

1. Elimina las imágenes previas que el programa tenga abiertas.
2. Desactiva el retículo, es un marcador que se sitúa sobre la coordenada introducida.
3. Desactiva el overlay, se trata de información proporcionada por Aladin, como, por ejemplo, el zoom aplicado.
4. Obtiene una imagen de la fuente POSSI.
5. Obtiene una imagen de la fuente POSSII.
6. Espera a que ambas imágenes se descarguen antes de continuar.
7. Utiliza la herramienta RGB sobre las vistas 1 y 2, las imágenes recién descargadas.
8. Espera a que se complete el proceso.
9. Aplica un zoom x4 sobre la coordenada usada como parámetro.
10. Realiza una espera para asegurar que el zoom se ha realizado correctamente.
11. Almacena la imagen resultante en la ruta indicada.

En varias instrucciones se emplea el valor $1, cada vez que esto aparece se está haciendo referencia al parámetro 1, en este caso la coordenada.

Las instrucciones get del script, marcadas en azul, obtienen una imagen de la fuente indicada, en nuestro caso empleamos:

- **POSSI** y **POSSII** para descargar imágenes con declinación positiva.
- **POSSI** y **SERC** para las negativas.

Las siglas POSS hacen referencia a **Palomar Observatory Sky Survey**, situado en San Diego, California. SERC representa **Space Environment Research Centre** situado en Australia.

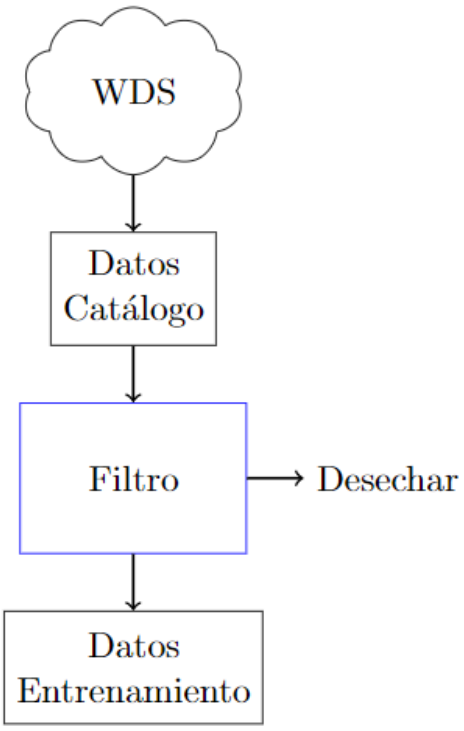


Figura 4.3: Diagrama de procesamiento para WDS.

4.3. WDS

La obtención de recursos de trabajo ya ha sido resuelta en la sección 4.2, sin embargo, es necesario obtener un listado de sistemas de estrellas dobles ya reconocidas que poder analizar con el fin de semi-automatizar su reconocimiento, figura 4.3.

WDS, Washington Double Star Catalog disponible en `http://ad.usno.navy.mil/wds/`, es un catalogo mantenido por el Observatorio Naval de los Estados Unidos que recopila información sobre sistemas de estrellas múltiples.

El problema es que el catalogo almacena gran cantidad de estrellas, y nosotros queremos casos de sistemas de estrellas dobles de movimiento rápido. Con el fin de obtener la información que nos interesa hemos creado un programa que filtra dichos datos basándose en los siguientes criterios:

- La última vez que se vio el sistema debe ser como mínimo 1975, esto asegura que en los catálogos existirán al menos dos imágenes, una relativamente actual y una de los años 50.

- La magnitud de las estrellas que componen el sistema deben ser a lo sumo 19, de este modo las estrellas serán apreciables a simple vista.
- La separación debe pertenecer al rango entre 2 y 180, las estrellas con valores mayores están demasiado alejadas y desvirtuarían los resultados.
- El desplazamiento ha de ser superior a 60 para que el movimiento sea apreciable.

Las coordenadas de todos los sistemas que sigan dichos criterios serán almacenadas en un fichero que puede usarse como parámetro de los scripts descritos en la sección 4.2.

4.4. Conjuntos de entrenamiento positivos y negativos

Usando las coordenadas obtenidas al filtrar el catálogo WDS, sección 4.3, como parámetro de los scripts de Aladin descargamos las imágenes que conforman conjunto de entrenamiento con datos positivos, esto es un conjunto de imágenes -ya catalogadas- con alto movimiento propio común. Para poder formarlo completamente se emplearon los generadores de coordenadas para obtener un número equivalente de casos en los que no hay estrella doble. Con el dataset completo se decidió aplicar una división común en machine learning que consiste en partir un dataset en tres, training, test y validation, correspondientes al 60 %, 20 % y 20 %, respectivamente, de los datos. De esta forma podemos evaluar el método, midiendo tanto los falsos positivos (cuando indica que hay una estrella doble y no la hay), como en los casos negativos (cuando descarta una imagen que sí tiene una doble).

En primer lugar tenemos el training set, estos datos serán los empleados para entrenar un modelo. El validation set se utiliza para afinar el valor de los parámetros del modelo generado. Por último, el test set es un conjunto de datos, independientes de los de entrenamiento, empleado para comprobar el rendimiento del modelo ya entrenado y afinado. Si los resultados obtenidos empleando el test set no son los deseados se debe volver a comenzar el entrenamiento desde cero con nuevos conjuntos de datos.

Esta técnica es muy útil debido a que si los datos de los training y validation están viciados, por ejemplo, si las estrellas dobles están formadas por componentes situadas a la misma distancia aparente, en pixels, dentro de la imagen, una vez se evalúe sobre el test set podremos comprobar que el modelo no es válido pues tiene en cuenta una distancia que no tiene por qué ser la misma siempre. Si los datos del test set también estuvieran viciados esta técnica no sería de utilidad.

Capítulo 5

Workflow

Resumen: Una vez resuelta la recolección de datos es necesario ver qué tratamientos aplicarles para alcanzar nuestro objetivo, pero antes vamos a definir una plataforma genérica sobre la cuál desarrollar dichos procesos.

5.1. Introducción

En los capítulos 6 y 7 se describen diferentes procesos por los que una fotografía ha de pasar para poder detectar sistemas de estrellas dobles. Cada una de estas etapas puede recibir un número arbitrario de entradas y salidas, además recibirá un evento de la clase Thread de Python que al abortar la ejecución, mediante la interrupción Ctrl+C, se activará permitiendo así a la etapa saber que debe parar su ejecución. Por ejemplo, la etapa de recoloreado que veremos en este capitulo recibe dos parámetros de entrada, uno de salida y un evento en el que poder consultar, mediante la instrucción "stop_event.is_set()" si debe parar de ejecutarse.

```
def run(in_lista, in_dimages, out_dout, stop_event):
```

Cada etapa por tanto puede ser configurada y lanzada por separado puesto que para conectarlas solo es necesario que su salida sea la entrada de la siguiente. De esta forma todas las etapas pueden trabajar de forma simultánea. Sin embargo, creemos que esta tarea puede simplificarse mediante la creación de un mecanismo de flujo de datos, que tiene la característica de que varios pasos se pueden ejecutar simultáneamente. En nuestro caso es posible conseguir esta concurrencia de tareas porque se asume que las etapas producen elementos de salida de forma constante que las siguientes etapas pueden consumir. Está filosofía recuerda a la empleada por la herramienta flume (`https://flume.apache.org/`) en entornos Big Data, empleada para recibir y operar con datos en streaming.

5.2. Estructura

La idea es generar una estructura que permita al usuario definir el orden de las etapas de manera sencilla, y una vez definidas las lance en diferentes hilos. Una posible representación del workflow sería un grafo acíclico dirigido, es importante que sea acíclico puesto que de otro modo la información nunca llegaría a un estado definitivo en el cual se acepte o rechace.

Podemos entonces definir cada fase como una entrada, una salida y un nombre de función, de este modo conectar una etapa con otra tan solo implica igualar la entrada de una a la salida de la otra. El nombre de función se almacena puesto que a la hora de lanzar el hilo es necesario saber a qué función llamar.

Existe un último, algunas etapas del workflow pueden tener más de una entrada o salida por lo que se debe emplear un array de entradas y otro de salidas, debido a esto también será necesario especificar el numero de entrada o salida a la que se hace referencia al conectar los pasos.

Una etapa se almacenara en el workflow empleando un diccionario, tendría por tanto el siguiente formato.

```
flow['step'] = {
  'input': [
    flow['other']['output'][0],
    settings.directory
  ],
  'output': [
    settings.directory2
  ],
  'callback': step.run,
}
```

Además se hace uso de una etapa *dummy* que tan solo define la entrada del primer paso en su salida. Una vez creadas las entradas en el diccionario para todas las etapas solo será necesario ejecutar el workflow.

5.3. Etapas

Si bien es cierto que el workflow esta pensado para dar libertad a la hora de implementar las etapas tiene dos requisitos que se han de cumplir. El primero es a la hora de definir el método que se utilizara como callback en el diccionario, sus parámetros deben estar ordenados de modo que reciba primero todas las entradas, luego todas las salidas y por ultimo un evento de las clase *threading*, este orden ha de ser tenido en cuenta a la hora de crear la entrada del diccionario descrita en la sección 5.2. En segundo lugar se debe asegurar que la etapa se mantendrá a la espera de nuevos datos que

procesar hasta que se active el evento que se recibe por parámetro y que una vez recibido no se abortara la ejecución del paso hasta que sea seguro.

Para poder saber qué está pasando el workflow también monta un sistema de log, utilizando la biblioteca *logging* de python, al cual las etapas pueden acceder para escribir mensajes de error o información de debug que le pueda ser de utilidad al usuario para conocer que ha ocurrido.

Debido a que las etapas se encuentran en continua ejecución podría darse el caso en el que los mismos datos se procesen varias veces. Para prevenir esto las etapas que hemos implementado mantienen su propio log de historia, por cada dato procesado se almacena un identificador, normalmente el nombre.

5.4. Uso

Para utilizar el workflow son necesarios tres pasos.

En primer lugar modificar el fichero de configuración, *settings.py*, en el cual se encuentran almacenados todos los directorios de los pasos que hemos creado y los parámetros de procesamiento que pueden modificarse de cada etapa.

Ir al workflow, *workflow.py*, y editar la función **define_flow()** para crear el diccionario de etapas tal y como se ha explicado en la sección 5.2.

Por último, ejecutar el workflow. Para parar el procesamiento basta con enviar una interrupción, Ctrl+C, en la consola en la cual se haya lanzado.

Capítulo 6

Procesamiento

Resumen: A lo largo de este capítulo se presentaran las diferentes etapas a las que se pueden someter los datos, entre ellas se encuentran transformaciones que actualmente no se emplean en nuestro detector pero que podrían ser útiles si se exploran nuevas formas de detección.

6.1. Recoloreado

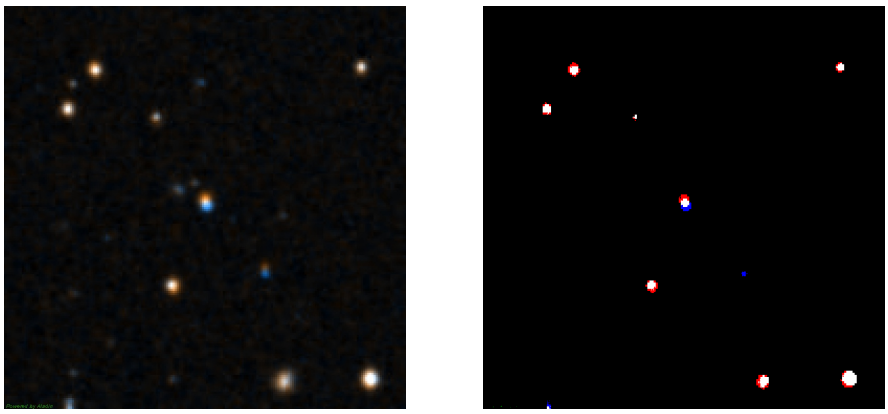
Esta etapa emplea la biblioteca PIL, Python Imaging Library, para cargar las imágenes y crear el canvas sobre el que guardar el resultado.

A simple vista las imágenes obtenidas en la sección 4.2 están coloreadas y nos permiten apreciar los contornos de las estrellas fácilmente. El problema surge cuando queremos que el programa haga las misma extrapolaciones que hace nuestro cerebro.

La utilidad de este proceso radica en que en una imagen pueden aparecer varios tonos de rojo, por ejemplo. Cuantos más tonos de un color aparezcan más difícil será ver los contornos. Con el fin de facilitar este reconocimiento la imagen pasa por un proceso de recoloreado que la transforma pixel a pixel que reduce el numero de tonos de cada color a uno.

Los colores que reconoce este proceso se definen mediante un diccionario que almacena el nombre y composición RGB del mismo. Esto facilita la modificación de los grupos reconocidos de modo que un cambio en el coloreado inicial de las imágenes no supone un problema.

```
groups = {
  'red': (255,   0,   0),
  'green': (0, 255,   0),
  'blue': (0,   0, 255),
  'black': (0,   0,   0),
```



(a) (b)

Figura 6.1: Recoloreado usando la distancia Manhattan

```
'white': (255, 255, 255)
}
```

El objeto de este proceso es definir cómo ha de interpretar la tonalidad de cada pixel de la imagen original, de forma que al finalizar esta fase se obtenga una imagen que contenga solo los colores definidos en el diccionario inicial. Para esto hemos empleado la noción de distancia conocida como "distancia Manhanttan". Se trata de medir la distancia entre la composición del pixel y la del grupo, asumiendo que ambas tonalidades son RGB y que están almacenadas en triplas, de la forma (R, G, B), tan solo hemos de restar cada componente con su homóloga y sumar los resultados para obtener un valor. El grupo que resulte con menor valor será aquel al que pertenezca el pixel.

Tal y como muestra la imagen 6.1b hay información que se pierde durante el proceso debido a que las estrellas están muy apagadas haciendo que la distancia al negro sea menor que al azul o rojo. Para poder atajar este problema se realiza una comprobación posterior, si el grupo asignado es negro pero la distancia al azul o al rojo es inferior a un cierto umbral se reasigna el grupo del pixel, imagen 6.2.

Para producir la imagen recoloreada creamos una imagen homóloga a la original sobre un canvas, pero el color de cada pixel se corresponderá al asignado durante el procesamiento.

6.2. Comprobación

Aún cuando el detector acepta un sistema de estrellas dobles existe un problema, puede no tratarse de un sistema nuevo. Para solventar esto exis-

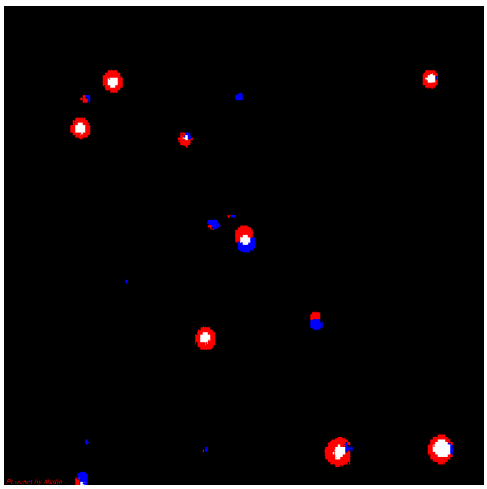


Figura 6.2: Recoloreado de la imagen 6.1a con un umbral de 150

te una fase de comprobación que se puede activar para todos los sistemas aceptados.

Si se incluye esta fase dentro del flujo de datos, se realiza un primer paso que consiste en conectarse a la url en la que se encuentra alojado el catálogo de WDS, `http://ad.usno.navy.mil/wds/Webtextfiles/wdsweb_summ.txt`, y descargar los datos de los sistemas conocidos, este paso solo se hace una vez por ejecución, al comienzo. El hecho de emplear los mismos datos durante una ejecución completa no supone un problema puesto que la base de datos no se altera habitualmente.

Cuando los datos de un sistema aparecen en la entrada de esta fase se toman sus coordenadas y se buscan coincidencias en todas las entradas descargadas, si no hay ninguna se ignora el sistema detectado. El resultado, es un fichero con el formato de intercambio de datos JSON de la siguiente forma.


```
{
  "1": {
    "Angle difference": 0.5332939070674456,
    "Separation difference": 6.23413572082471,
    "Maximum separation": 117.64777940955791,
    "Separation %": 5.298982906530099,
    "PA": 151.86140052006294,
    "Separation": 28.995116847884947,
    "Proper Motion A (brightest)": [
      163.2,
      -258.4
    ],
```

```
    "Proper Motion B": [
      176.79999999999998,
      -299.2
    ]
  }
}
```

El fichero almacena los siguientes datos:

- *Angle difference*: es la diferencia entre los ángulos de posición formados por el "sistema de estrellas rojas" y el "sistema de estrellas azules", es decir, indica la evolución del ángulo de posicion.
- *Separation difference*: es el resultado de restar la diferencia entre las separaciones de las "estrellas rojas" y "azules" del sistema.
- *Maximum separation*: es la mayor de las dos separaciones empleadas en el dato anterior.
- *Separation %*: es el tanto por ciento de la *Maximum separation* correspondiente al valor de *Separation difference*.
- *PA*: es el valor medio de los ángulos de posición formados por las "estrellas rojas" y "azules" del sistema.
- *Separation*: es la media de ambas separaciones multiplicada por un factor de conversión, 1/3.95, que normaliza el valor al sistema empleado por WDS.
- *Proper motion*: es el movimiento própio de cada estrella multiplicado por un factor de conversión, 6.8, que normaliza el valor.

Si por el contrario existen datos asociados al sistema se almacenan en el archivo JSON junto con los datos proporcionados por el detector y el cálculo del error entre ambos, dando lugar a un archivo con el siguiente formato.

```
{
  "1": {
    "Angle difference": 1.419208447017608,
    "Separation difference": 3.3066248088095165,
    "Maximum separation": 123.22743201089601,
    "Separation %": 2.683351226955,
    "PA": 15.760824216099781,
    "Separation": 30.77825812822563,
    "Proper Motion A (brightest)": [
      136.0,
      0.0
    ],
```

```
    "Proper Motion B": [
      122.39999999999999,
      27.2
    ]
  },
  "wds": {
    "PA": 18.0,
    "Separation": 30.55,
    "Proper Motion A (brightest)": [
      93,
      15
    ],
    "Proper Motion B": [
      91,
      10
    ]
  },
  "error": {
    "PA": 2.239175783900219,
    "Separation": -0.22825812822562952,
    "Proper Motion A (brightest)": [
      -43.0,
      15.0
    ],
    "Proper Motion B": [
      -31.39999999999999,
      -17.2
    ]
  }
}
```


Si bien esta fase es una primera comprobación sus resultados no son definitivos en el caso del no, puesto que solo se comprueban las coordenadas centrales de la imagen. Si la estrella no se encuentra en el centro es imposible para este programa determinar si el sistema ya se conocía o no.

6.3. Contador de pixels

Al igual que la etapa definida en la sección 6.1 se emplea la biblioteca PIL. Esto no es lo único que comparten, el proceso al que se someten las imágenes es muy similar.

La idea de aplicar este proceso es obtener la composición de colores de cada fotografía. Una vez obtenida se almacena el resultado en un fichero CSV que se puede analizar posteriormente. La gama de colores se define mediante un diccionario en el que se almacenan los colores puros que se quieren reconocer.

Una vez cargada la imagen se analiza cada pixel que la compone, para conocer el color, de entre los definidos en el diccionario, al que más se asemeja. como el nombre de la etapa indica el objetivo es contar los pixels, por lo tanto existe un contador por cada color en el cual se almacena el número de pixels de la imagen que pertenecen a dicha tonalidad.

Al terminar de procesar es posible guardar el valor de dichos contadores o transformarlos para obtener otros datos. En nuestro caso, decidimos almacenar los porcentajes de pixels rojos, azules y blancos, así como las proporciones tanto de blancos como de azules con respecto de los rojos.

Esta etapa ya no se incluye en el workflow, esto se debe a que el objetivo era proporcionar al filtro descrito en la sección 8.1 los datos que necesitaba.

6.4. Corte

En esta etapa empleamos la biblioteca image_slicer de Python, distribuida bajo licencia MIT. Esta biblioteca permite dividir una fotografía en n piezas del mismo tamaño.

En este caso decidimos que las imágenes se dividieran en 9 sectores, lo cual permite reducir la información en la fotografía pero dejando datos suficientes en ella para posteriores análisis. En la figura 6.3 se puede observar que existe mucha información, sin embargo, si la dividimos en 9 fragmentos, figura 6.4, las imágenes resultantes tienen menos información facilitando la toma de decisiones sobre ellas.

Este proceso tan solo almacena el cuadrante central para reducir la carga de memoria, debido a esto solo la imagen de la figura 6.4e aparecería en la salida. Si esto quisiese modificarse tan solo habría que desactivar el parámetro ónly_center'del método run. Está etapa esta pensada para ser la primera del workflow, puesto que al eliminar trozos de la imagen carecería de sentido aplicar transformaciones previas que no aportarían nada.

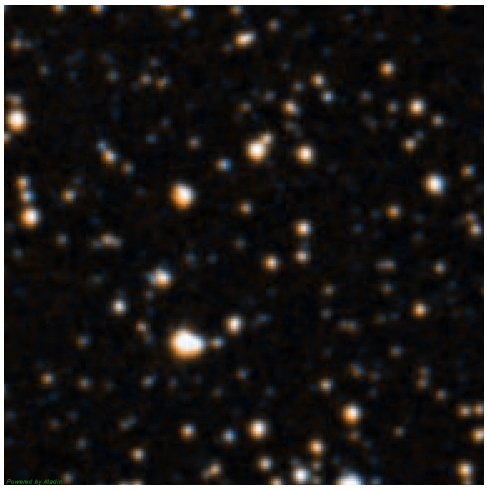
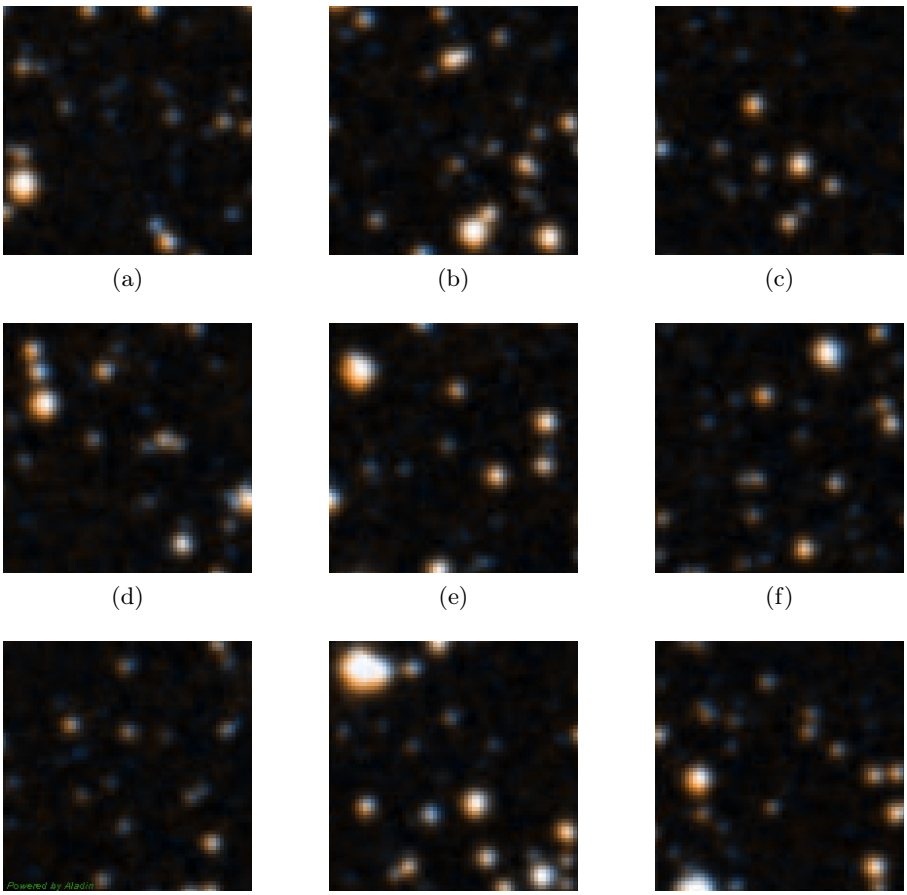


Figura 6.3: Imagen sin recortar



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 6.4: Imágenes resultantes de la división

Capítulo 7

Detector

Resumen: La detección es el paso que sigue al recoloreado y está dividida en dos partes. La primera parte se basa en el análisis de la imagen obtenida en el recoloreado. Este análisis se lleva a cabo separando la imagen en dos que representan los cambios a lo largo del tiempo y calculando las diferencias entre cada posible pareja de estrellas. Con los resultados de este análisis se procede a la fase de decisión la cual se encarga de procesar los resultados de la etapa posterior y decidir, para cada pareja de estrellas, si forman o no una estrella doble.

7.1. Análisis

Para el desarrollo de este fase, partimos de las imágenes descargadas por Aladin, que se encarga de superponer las imágenes de las dos fotografías aplicando un filtro de color a cada una, de manera que en la primera se reemplaza el blanco por rojo y en la segunda por azul. En la fase de recoloreado se ha potenciado el color de la imagen para que cada pixel sea un color puro, lo que facilita su tratamiento. Una vez obtenidas las imágenes, El objetivo es relacionar cada ?estrella roja? con una ?estrella azul? de forma que seamos capaces de detectar que son, en efecto, dos imágenes de una misma estrella. Si encontramos dos estrellas que se hayan movido en la misma dirección y sentido, y a una velocidad similar, podemos señalar la pareja como una candidata a estrella binaria.

Hemos programado esta fase en el lenguaje Python, debido a la gran cantidad de bibliotecas que permiten tratar imágenes.

En particular, para la detección de estrellas dobles se decidió utilizar la biblioteca de tratamiento de imágenes OpenCV. El desarrollo de esta fase se realizó en varias etapas.

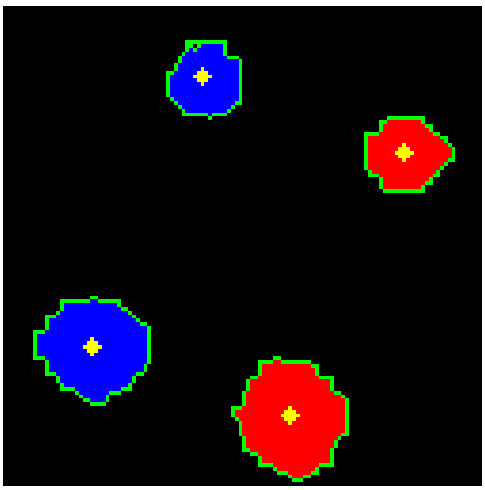


Figura 7.1: Estrella binaria con movimiento veloz

Después de analizar cientos de imágenes de estrellas dobles, se decidió recrear el mismo procedimiento que se realiza mentalmente a la hora de decidir si son dobles o no. Para decidirlo hay que fijarse en que sus trayectorias sean paralelas y la distancia recorrida sea aproximadamente la misma.

Para llevar a cabo este procedimiento necesitamos encontrar un centro aproximado de cada una de las estrellas del recoloreado para poder así buscar relaciones, de distancia y ángulo, entre parejas de estrellas cuyo movimiento haya sido similar a lo largo de los años.

Una primera aproximación se basó en la búsqueda de los contornos de las diversas estrellas en función de sus colores y el uso de estos contornos para obtener el centro de la estrella. Esta aproximación permitió relacionar aquellas estrellas cuyo movimiento hubiese sido lo suficientemente rápido como para que no compartan ningún punto entre la primera foto y la segunda.

El movimiento de la estrella de la figura 7.1 es suficientemente rápido como para que entre la primera foto y la segunda no exista ningún punto en común por lo que la obtención del centro para su posterior análisis no requiere de ningún cambio en la foto. El problema de este método radica en que si la estrella no se ha movido lo suficientemente rápido, existirán puntos comunes, representados en color blanco. Debido a esto, se obtendrían tres centros distintos para una misma pareja de estrellas y ninguno de ellos serían los centros reales de las dos estrellas.

Por tanto, si la estrella no se ha movido lo suficientemente rápido. Se obtendrían tres centros distintos para una misma pareja de estrellas y ninguno de ellos serían los centros reales de las dos estrellas.

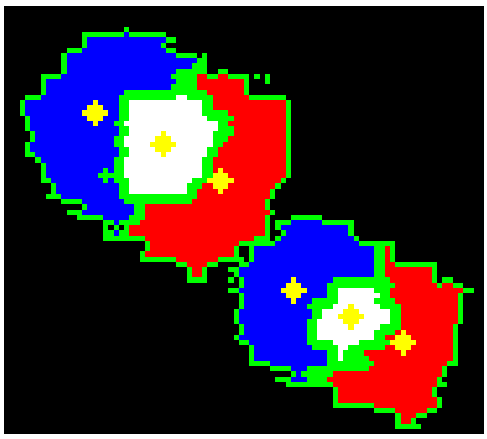


Figura 7.2: Estrella binaria cuyo movimiento no es tan apreciable debido a su menor velocidad

Para subsanar el error se decidió realizar el análisis de contornos por separado. Para ello se filtran la imagen en dos imágenes de manera que una tiene el color rojo y el blanco, y la otra el azul y el blanco. De esta manera se obtienen los centros y áreas de las estrellas en origen y los centros de las estrellas en el destino y se pueden relacionar de manera directa. Esto no se puede hacer directamente con las imágenes originales ya que habría que escalar y girar las imágenes de manera que coincidieran. Esta tarea la realiza Aladin con gran precisión.

Una vez obtenidos los centros y áreas de todas las estrellas se procede a especificar cuál sería la estrella primaria, la de mayor área, y cuál sería la estrella secundaria, de menor área. Y calculamos los valores necesarios para su clasificación, figura 7.5.

Los valores representados en la figura 7.5 corresponden a:

(a) Separación entre las estrellas en la segunda foto tomada.

(b) Distancia recorrida por la estrella secundaria.

(c) Separación entre las estrellas en la primera foto tomada.

(d) Distancia recorrida por la estrella primaria.

(e) Ángulo de la trayectoria de la estrella secundaria.

(f) Ángulo de la trayectoria de la estrella primaria.

(g) Ángulo de la posición de la estrella secundaria con respecto a la primaria en la primera foto.

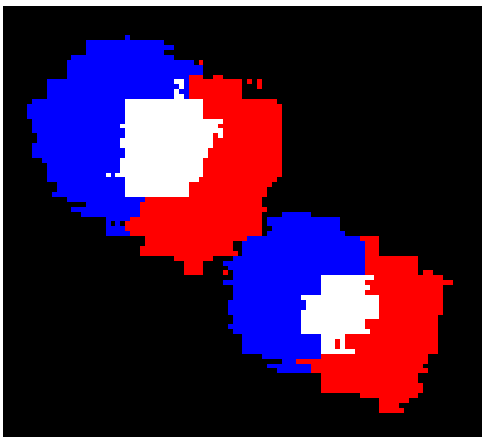


Figura 7.3: Estrella binaria con bajo movimiento propio

(h) Ángulo de la posición de la estrella secundaria con respecto a la primaria en la segunda foto.

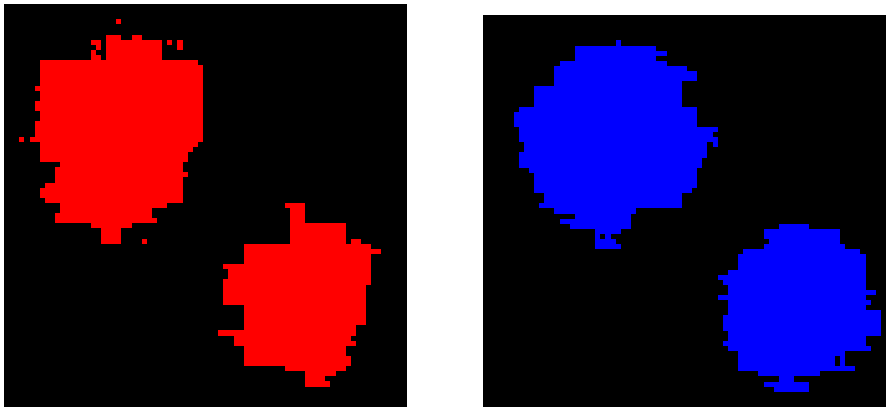
Una vez obtenidos los datos de la figura 7.5 se procede a procesarlos para calcular los valores que se utilizan para clasificar estrellas dobles.

7.2. Decisión

Una vez obtenidos los valores de la sección 7.1 se procede al estudio de los mismos, para decidir si la pareja de estrellas es binaria o no. Previamente al diseño de nuestro algoritmo, se estudió qué similitudes hay entre las parejas de estrellas binarias ya catalogadas. Se llegó a la conclusión que estas mantienen entre sí su posición relativa a lo largo de los años. Es decir, la separación entre ellas, el ángulo de la una con la otra, el ángulo de su trayectoria debería ser aproximadamente iguales. Además, la relación entre el área de cada una de las estrellas debe mantenerse a lo largo del tiempo.

Una vez decididos valores necesarios para tomar la decisión procedimos a establecer unos valores críticos que marcaran el límite entre lo que podría considerarse estrella doble y lo que no. En caso de que la diferencia entre los valores antes y después, por ejemplo la distancia entre las estrellas, fuera superior al valor crítico decidido para la distancia, la estrella automáticamente sería descartada. En caso contrario se pasaría a comprobar el siguiente valor. Estas comprobaciones actuarían como un filtro por etapas, de manera que si la pareja de estrellas supera todas las etapas, sería considerada estrella doble.

Estos valores críticos son:



(a) (b)

Figura 7.4: A partir de la imagen recoloreada, figura 7.3, se obtienen dos imágenes

- El error entre los ángulos que forman ambas estrellas puede ser como máximo 5.
- El error entre los ángulos del vector velocidad es como máximo 15.
- La diferencia entre sus desplazamientos o separaciones no puede sobrepasar el 7 %.
- La proporción entre el area de la componente principal y la secundaria del sistema es, a lo sumo, 2.
- La proporción entre las distancias recorridas no puede ser superior a .1.3
- Una estrella no puede haber realizado un desplazamiento superior a 80.
- La separación de cualquier componente del sistema no puede ser superior a 200.
- Debido que los sitemas binarios no son comunes y puesto que los cúmulos de estrellas generan problemas, no puede existir más de una estrella doble.
- Si la fotografía tiene más de 60 estrellas o bien se trata de un cúmulo o bien de un fallo de superposición por lo que no es válida.
- El movimiento propio de ambas componentes ha de ser inferior a 2500, es tan inusual encontrar valores mayores que casi con total seguridad se trataria de un error.

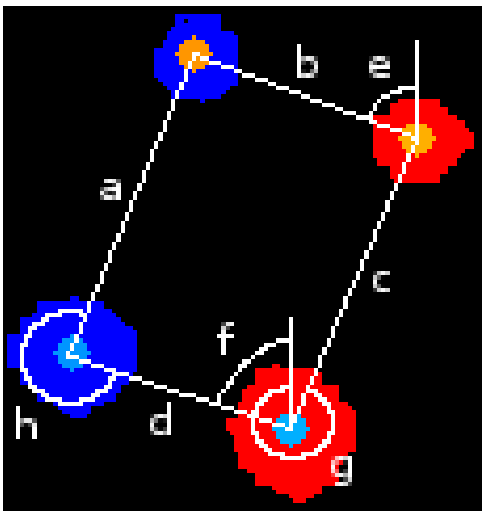


Figura 7.5: Representación de valores calculados

Con el fin de normalizar los datos al formato de WDS, se calcularon los factores de conversion que hay que aplicar a nuestros cálculos:

- El movimiento propio se ha de multiplicar por 6.8.
- La separación se multiplica por 1/3.95.

Además de los filtros anteriormente descritos, se añadió un filtro más de manera que si la fotografía contuviese más de una pareja de estrellas, lo cual es bastante difícil que se de, se descarta porque probablemente se deba a que el montaje de las dos fotografías está mal hecho, por los motivos mencionados en la sección 4.1, y parece que todas las estrellas se desplazan en la misma dirección o porque la fotografía representa un cúmulo de estrellas en los cuales nuestra aplicación no podría detectar nada.

Una vez que la estrella ha pasado todos los filtros, se procede a pintar sus centros de otro color en la imagen recoloreada y se guardan los valores de sus atributos obtenidos de su análisis, de manera que sea fácil su comprobación.

Capítulo 8

Problemas

Resumen: En este capítulo vamos a hablar de las ideas y procesos que han sido descartados del programa final.

8.1. Filtrado por recuento y proporción de colores

8.1.1. Introducción

Esta fase consistía en realizar un modelo de Regresión Logística que clasificara imágenes en dos clases: la que tiene estrellas dobles, que de ahora en adelante nos referiremos a ella como "clase del sí" y las que no, que de ahora en adelante nos referiremos a ella como "clase del no".

La regresión logística, a pesar de su nombre, es un modelo lineal para la clasificación en lugar de la regresión. En la literatura el nombre más común por el que se encuentra a la Regresión logística el de "logit".

El motivo por el cual decidimos utilizar la Regresión Logística, es debido al buen comportamiento que desempeña en los problemas de clasificación binarios.

Para el desarrollo de esta sección, que tiene que ver con Machine Learning, hemos utilizado la biblioteca Scikit-learn que nos proporciona Python, dado que es una de las más usadas, completas y mejor documentadas que hay. Pedregosa et al. (2011)

8.1.2. Idea principal

Este modelo de regresión logística estaba pensado para realizar predicciones sobre la información que proporcionaba la fase del conteo de píxeles, el color de los píxeles de la imagen, decidiendo si había o no una estrella

doble. La información que utilizábamos de cada imagen era el porcentaje de rojo, azul, blanco y la división entre el porcentaje de azul y el de rojo, la del blanco entre el del rojo y si existía o no una estrella doble en la imagen.

Al modelo se le entrena con un conjunto de datos pensado para realizar solamente el entrenamiento, las predicciones se realizan con otro conjunto de datos distinto llamado conjunto de validación y en base a estas predicciones se afinan a mano los distintos parámetros del modelo de Regresión Logística para mejorar la precisión de las mismas; por último una vez se hubieron ajustado al máximo los parámetros del modelo y las predicciones son lo suficientemente buenas, se realizan otras predicciones sobre otro conjunto de datos distinto llamado conjuntos de test, sobre el cual se verifica que la precisión de estas predicciones son igual de buenas que con el conjunto de validación. Si estas predicciones no fuesen tan buenas se debería realizar otra vez el proceso de separación en los distintos conjuntos y empezar de nuevo con el entrenamiento del modelo.

Antes de continuar con esta exposición es necesario aclarar unos conceptos que aparecen a la hora de realizar las predicciones, estos son la precisión y la sensibilidad:

- La precisión es la fracción de instancias relevantes entre las instancias recuperadas.
- La sensibilidad es la fracción de instancias relevantes que se han recuperado sobre el total cantidad de instancias relevantes.

Por ejemplo, supongamos que nuestro programa predice que 5 imágenes tienen estrellas dobles de un conjunto con 7 imágenes con estrellas dobles. De las 5 imágenes identificadas sólo 3 tienen en realidad estrellas dobles y el resto no. La precisión sería entonces 3/5, mientras que la sensibilidad sería 3/7. Una vez aclarados estos conceptos podemos seguir con la exposición.

La intención que teníamos era entrenar el modelo de tal manera, que nos predijera el caso del no una precisión casi absoluta consiguiendo además una sensibilidad lo suficientemente alta, de manera que para la siguiente fase ya se habrían descartado la mayor cantidad de imágenes en las que no existiera una estrella doble.

Después de entrenar el modelo centrándonos en satisfacer estas restricciones observamos que está primera aproximación resultó funcionar extraordinariamente bien, superando incluso nuestras expectativas alcanzando una precisión y una sensibilidad que llegaban al 99 %. Pero esto era solo una ilusión, puesto que el programa no funcionaba tan bien como podía parecer a simple vista.

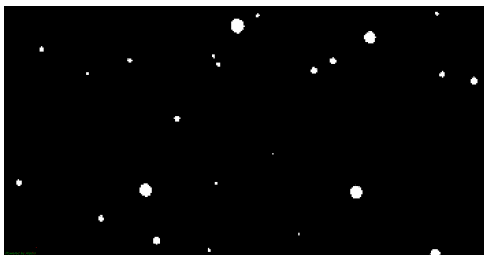


Figura 8.1: Ejemplo de imagen en la que no hay una estrella doble.

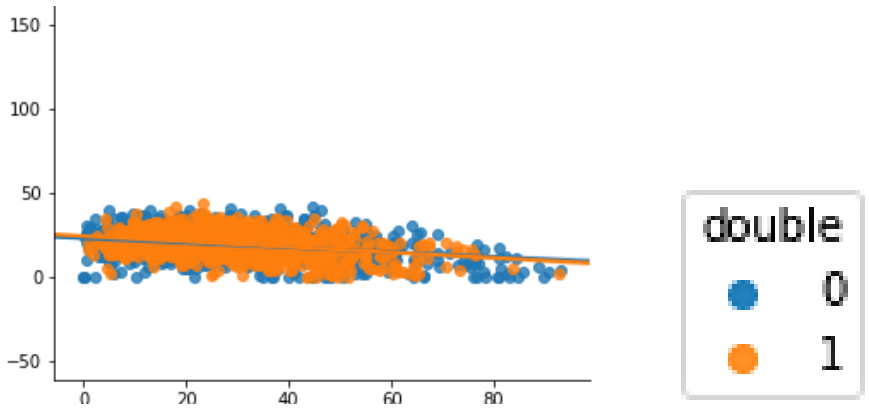
8.1.3. Ilusión de funcionamiento correcto

En primera instancia, el modelo parecía funcionar realmente bien y no fue hasta que probamos el modelo junto al resto del workflow, cuando nos dimos cuenta que, como se ve en la figura 8.1 la gran mayoría de las imágenes que el programa había predicho que no tenían estrella doble sólo tenían todas las estrellas blancas exclusivamente. Aunque es cierto que todas las imágenes que son como la figura 8.1 no tienen estrella doble, esto no quiere decir que ninguna de las imágenes con colores rojo y azul no fueran imágenes sin estrellas dobles, y lo que ocurría entonces es que, lo que acababa aprendiendo el modelo era que si la imágen tiene una cierta cantidad de blanco en esa imagen, entonces no hay estrella doble.

Una vez hubimos identificado el problema nos dispusimos a cambiar el programa de tal manera, que si la tras realizar el conteo de píxeles una imagen tenía una cantidad extremadamente alta de blanco, esta se eliminaba antes de llegar a esta fase y así evitar los problemas con el modelo. Una vez se hubieron "limpiado" el conjunto de imágenes nos dispusimos a realizar de nuevo al los pasos de entrenamiento del modelo, validación y test con los nuevos conjuntos y comprobar cómo se comportaba el modelo a la hora de realizar las predicciones.

8.1.4. Causas del descarte

A pesar de la limpieza que realizamos en los conjuntos, el siguiente problema que encontramos es que el modelo no realizaba buenas predicciones para ninguna de las dos clases, la precisión era menor del 60 %. Nuestro modelo no era capaz de saber si en una imágen había o no una estrella doble sólamente con la información del color de sus píxeles. Está conclusión aunque decepcionante tampoco era del todo sorprendente, en la figura 8.2 se ve la relación que hay entre los atributos de píxeles blancos y los azules y si hay o no una estrella doble, se puede apreciar perfectamente que no hay una división posible en la que puedas separar las dos clases de una forma clara y coherente. Este es sólo un ejemplo pero este problema se da entre cada



(a) (b)

Figura 8.2: En el eje x pertenece al atributo “ %blue”, el eje y pertenece al atributo “ %white”.

par de atributos y es una de las razones por las que las predicciones no son buenas.

Entonces encontramos una documentación muy extensa y valiosa sobre una de los 16 parámetros de los que dispone el la constructora del modelo de Regresión Lógica : “class_weight”. Con este parámetro se puede variar el peso que tiene cada una de las clases y dar así más importancia a unas determinadas clases y menos a otras. Esto es realmente útil cuando se trabaja con modelos que están desbalanceados, y aunque en este caso nuestro modelo no tenía esta peculiaridad pensábamos que si dábamos más importancia a la clase del no, podríamos conseguir una mayor precisión en las predicciones de esta clase a costa de una pérdida en la sensibilidad. Y se consiguió que la precisión aumentara a un 83 % pero la sensibilidad disminuyó a un 2 %.

Aunque el comportamiento final del modelo conseguimos que mejorara, no cumplía con nuestros requisitos de poder descartar con su uso un gran número de imágenes que no tuvieran estrellas dobles, con lo que finalmente decidimos descartar su uso.

8.2. Parametrización automática

El siguiente paso consistía en procesar todos los datos calculados en la fase de análisis con la ayuda de un modelo de Machine Learning para encontrar el ajuste de los valores que hacen únicas a las estrellas dobles frente al resto de estrellas.

La existencia de este paso se debe a que en primera instancia realizábamos a mano el ajuste de valores para decidir cuándo dos estrellas formaban parte de un sistema binario y cuándo no.

Debido a que este proceso era muy largo y tedioso, se nos ocurrió que podíamos implementar un programa que encontrara por nosotros cuáles eran esos valores que mejor ajustaran la clasificación de las estrellas dobles.

Por todo ello decidimos realizar un modelo de Machine Learning en Python con la ayuda de la biblioteca de Scikit-learn, dado que es la mejor para realizar este tipo de tareas.

En esta fase, nuestro trabajo consistió en realizar un proceso similar al de detección, pero con la particularidad de que los datos calculados para cada estrella en la imagen pasarían a ser introducidos en un modelo de Machine Learning basado en árboles de decisión conocido como "Random Forest", del mismo modo incluimos una clase que representa si la estrella analizada formaba parte o no de un sistema binario, de tal manera que pudiéramos aprovechar el árbol resultante para introducir en la fase de detección los valores concretos en la comprobación de si una estrella cumple con lo especificado en un sistema binario.

Para realizar este trabajo, el modelo propuesto debería analizar un conjunto de imágenes en las que hubiera en cada una de ellas estrellas dobles. Además deberíamos identificar a las dos estrellas que formaban parte del sistema binario.

Para conseguir esto descubrimos que si utilizábamos los atributos del ángulo de posición y la separación, teniendo en cuenta un margen de error, los podríamos comparar con los que nos proporciona el catálogo de WDS para la coordenada en la que está situada el sistema binario, y añadir así, la clase de estrella doble a las estrellas de la imagen que cumplan esa condición, que no deberían ser más que dos.

Durante la implementación de esta fase investigamos acerca de cómo obtener un árbol de decisión a partir del modelo generado, que nos permitiera obtener un criterio a la hora de discernir sobre si dos estrellas pertenecen o no a un sistema binario.

En este momento nos dimos cuenta que, aunque los modelos basados en árboles de decisión, como Random Forest en nuestro caso, permiten obtener dicho árbol, este no nos proporcionaba la información que buscábamos, en cambio, nos facilitaba un árbol por cada estimador del modelo que en cada nodo indicaba si las muestras pasaban por él o no.

Posteriormente encontramos en la clase de árboles de decisión, concretamente en la clase "Decission Tree Classsifie", una funcionalidad propia de ésta que, utilizando el módelo generado, devuelve un archivo con extensión ".dot" en el que se encuentra la representación del árbol de decisión generado por el modelo.

Una vez generado el archivo.dot, con la ayuda de una biblioteca de Python utilizada para dibujar grafos llamada "graphviz", se puede obtener la

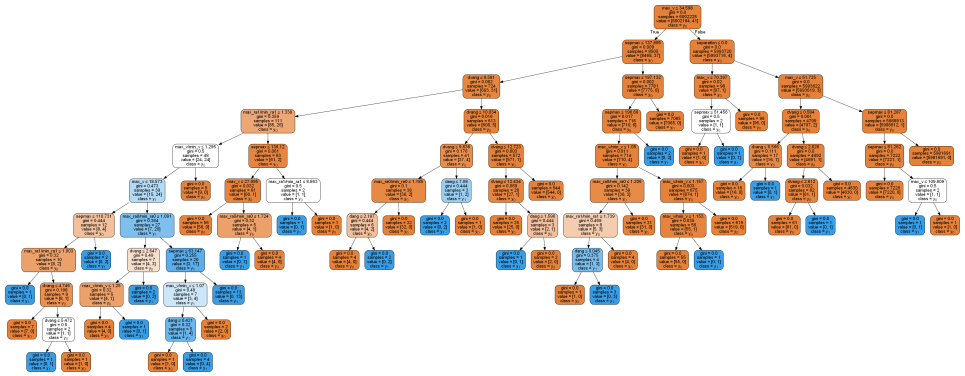


Figura 8.3: Árbol de decisión generado a partir del modelo.

representación gráfica del árbol, con la información de el valor del atributo utilizado para decidir en cada nodo y el valor de la clase en los nodos hoja, figura 8.3.

Aunque esta era justo la información que buscábamos, esta sólo se podía analizar a mano, puesto que el formato del archivo que utiliza esta biblioteca para dibujar el árbol no es más que un string que representa a la imagen.

Por otro lado el modelo obtenido no era del todo satisfactorio puesto que aunque la precisión para la clase del no era muy buena llegando al 100 %, no lo era para la clase del no, llegando sólo a un 20 %.

Debido a este contratiempo en el último momento del desarrollo, que no pudimos subsanar, decidimos simplemente descartar esta fase de parametrización y trabajar con el ajuste que se realizó a mano en las primeras instancias de la detección.

Capítulo 9

Conclusiones

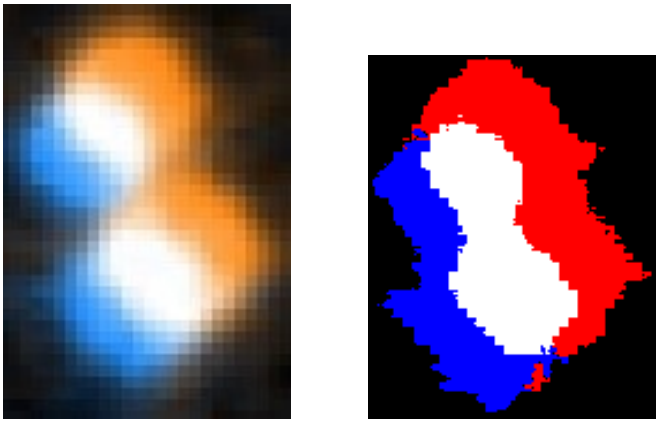
Resumen: En este último capítulo vamos a ver los resultados obtenidos y analizar el motivo de los mismos.

Como ya adelantó el título de este proyecto, el sistema es semi-automático debido a que requiere del apoyo externo de una persona con experiencia suficiente en la detección de estrellas binarias. El proyecto se diseñó de tal manera que, dada una foto, fuera capaz de decidir con casi un 100 % de acierto que, si no había una estrella doble, dijera que no. Esto se decidió debido a que lo más importante era no perder ninguna estrella, aunque esto provocará una mayor carga de trabajo a la hora de revisar las fotografías en las que nuestra aplicación detectara una estrella doble.

Los resultados de nuestro sistema son un 97 % de acierto en las imágenes en las que no existe ninguna estrella binaria y un 33 % de acierto a la hora de encontrar una estrella binaria en las que sí exista una. Volviendo a la idea inicial que mencionamos, hemos obtenido un resultado satisfactorio cuando se trata de rechazar fotografías que no contienen esta tipología de sistemas, un porcentaje de acierto en el no tan elevado reducirá en gran medida la carga de trabajo de los expertos.

En lo que a los sistemas binarios se refiere, el 33 % de los marcados contienen realmente una doble esto es debido a una decisión inicial en el diseño del sistema. Concretamente a que la detección se realiza con imágenes que han sido modificadas en la fase de recoloreado. En la cual los colores se llevan a su máxima intensidad por lo que si dos estrellas están muy próximas, al recolorear, nuestro sistema las detectará como una estrella única, figura 9.1. Los sistemas binarios con estrellas tan cercanas son muy comunes y, por tanto, el porcentaje de aciertos se ve drásticamente reducido.

Como posible medida que se podría tomar para incrementar el porcentaje de acierto se encuentra un cambio al sistema de detección de centroides, de



(a) (b)

Figura 9.1: Comparativa fotografía original y recoloreada

manera que se realizase un estudio de la intensidad y densidad de los pixels en la fotografía, y se determinase con estos datos que conforma un centro de estrella. De esta manera ejemplos como el de la figura 9.1a podrían estudiarse como dos estrellas separadas, puesto que entre ambas es posible apreciar a simple vista un cambio de intensidad, y se vería que, en realidad, forman un sistema binario.

Realizar el estudio de las fotografías por separado manualmente sin que Aladin haga la combinación sería otra de las posibles mejoras de este proyecto. Para realizar la combinación es necesario alinear y escalar las fotografías para, una vez preparadas, realizar su análisis por separado en busca de estrellas dobles.

Capítulo 10

Conclusions

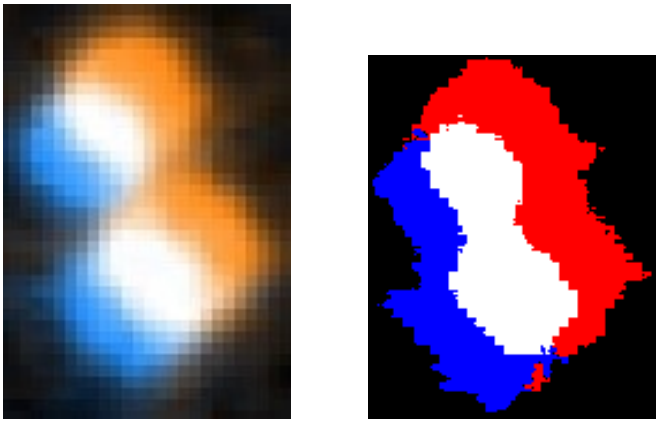
Abstract: In this last chapter we are going to discuss the results and analyze the reason behind them.

As already advanced by the title of this project the system is semi-automatic because it needs external support from a person with enough experience in the detection of binary stars. The project was designed to, given a picture, be able to determine with almost a 100 % accuracy that, if there was not a double star, the sector does not contain a double star. The reason behind this decision lays in the fact that the top priority for us is not to lose any star, even though this will mean a higher workload at the time of reviewing the photographs in which our application detected a possible double star.

The results of our system are a 97 % success rejecting pictures with no binary star present and a 33 % of the pictures containing a possible double actually had one. Going back to our initial idea, this outcome is satisfying when it comes to refuse pictures that do not contain this type of systems, such a high percentage of success in the no will indeed lighten up the amount of work required.

As far as binary systems are concerned, the 33 % of the systems detected as possible doubles were indeed one, this is because of a design decision. To be precise the problem is the intensity of the stars in the recolored picture in which colors are intensified so if two stars are very close the detector will only notice one, 10.1. This kind of close binary systems are pretty common so the success rate is drastically reduced.

A possible measure to prevent this kind of problem in the future, and consequently the success rate, would be to change the centroid detection. This new method would conduct a study of pixel intensity and density in the picture, and use this data to decide what really is the center of a star. In



(a) (b)

Figura 10.1: Comparative original vs. recolored

this way systems such as the one see in the example, figure 10.1a, could be studied as two separate stars since between both it is possible to appreciate an intensity variation at first sight, and it would be seen that, in reality, they form a binary system.

Performing the study of the photographs separately by hand without Aladin making the combination would be another possible improvement of this project. To make the combination it is necessary to align and scale the photographs so that, once prepared, the analysis can be performed separately in search of double stars.

Capítulo 11

Aportaciones

Resumen: En este capítulo cada integrante del grupo va ha exponer su aportación al trabajo y el desarrollo de esta.

11.1. David

Lo primero que necesitábamos para poder empezar a trabajar era obtener los sets de entrenamiento, así que comencé investigando el catálogo de estrellas dobles, WDS, tanto los datos que almacena para hacernos una idea de que íbamos a encontrarnos como el formato que les dan. Con estos dos puntos claros cree el programa que se conecta al catálogo para descargar sus datos y obtiene la lista de coordenadas de las estrellas cuyos parámetros se ajustan a los deseados. Con este listado ya generado creamos el script para descargar fotografías de coordenadas positivas.

Puesto que ya teníamos las componentes positivas del set de trabajo, investigue los distintos tipos de coordenadas celestiales que se emplean y me decanté por el modelo ecuatorial, una vez decidido el sistema de representación de coordenadas, hablé con mis compañeros y creamos ambos generadores, aleatorio y continuo, que utilizamos para obtener un fichero de coordenadas aleatorias. Con este fichero descubrimos que no se podían descargar coordenadas negativas con el primer script e hicimos un segundo que nos permitiese obtenerlas de otros repositorios.

Cuando teníamos las imágenes descargadas nos dimos cuenta de un fallo en los sets de entrenamiento, hay imágenes que no se pueden descargar bien sea porque el repositorio está caído o porque no tiene recursos almacenados para esa coordenada, esto daba lugar a fotos sin superponer o archivos corruptos puesto que mediante Aladin no se pueden gestionar estos errores. Para solventarlo limpié los sets a mano. Al mismo tiempo estaba desarrollando el algoritmo que contaba y recoloreaba los pixels de las fotografías, en este

punto se hacían ambos procesos a la vez y no por separado, tuve problemas para decidir cómo determinar el color de los pixels pero por suerte recordaba una medida de la distancia que, al menos en nuestro grado, aprendemos en inteligencia artificial, la distancia Manhattan. Se me ocurrió aplicarla a este problema concreto puesto que lo que quería determinar realmente era la distancia entre dos colores, el que tiene el pixel y los colores puros a los que se puede atribuir, tan solo me fue necesaria una pequeña modificación para intentar .ªclarar"los pixels, es decir, que cuando dijera que el pixel era negro compruebe aún así si podría ser azul o rojo. Como esta etapa la desarrollé con la idea de que fuera el primer proceso al que se sometieran las imágenes decidí que sería el encargado de limpiar las fotos cuya descarga sea errónea.

Mi compañero Javier estaba aplicando regresión logística sobre los datos que la etapa de recoloreado/conteo proporcionaba, sin embargo, el recoloreado no le aportaba nada a su programa por lo que se estaban duplicando la cantidad de imágenes almacenadas sin motivo alguno, fue en este punto donde decidí separar ambas funcionalidades, este es el motivo de su similitud, la etapa de recoloreado no se desechó debido al trabajo de Daniel, su programa necesitaba fotografías con un menor numero de canales, es decir, menos deferencias en las tonalidades de los pixels.

En este punto las etapas no funcionaban de manera fluida, era necesario que la etapa previa hubiera procesado todos los datos de entrada antes de que la siguiente se pusiera en marcha, esto era claramente un mal diseño. Ideé un primer concepto del workflow que ahora se emplea, en este punto las etapas se añadían creando manualmente el thread que las controlan, pero, como me hizo ver nuestro director, el tener que programar el thread oscurecía la estructura que se le daba al grafo de trabajo. Con la simpleza como objetivo modifiqué el formato, dando lugar a la estructura explicada en esta memoria gracias a la cual no es necesario tener ningún conocimiento del funcionamiento del workflow. Tan solo es necesario saber qué etapas se quieren y como se interconectan. Además del desarrollo me encargue de adaptar las etapas que ya estaban programadas y todas las que se crearon a posteriori a un formato estándar. También le añadí a todas la capacidad de usar un logging para informar en todo momento del estado de la ejecución.

Volviendo al trabajo de mi compañero Javier surgió un problema con las predicciones que realizaba el modelo de regresión logística, en un primer momento lo achacamos al zoom en los sets de trabajo y, por tanto, los volvimos a descargar pero esta vez sin zoom. A pesar de esto seguía fallando y ante la posibilidad de que hubiera demasiado ruido.[en] las fotografías cree la etapa crop para reducir tan solo el tamaño de las fotos, pensamos que este recorte podría funcionar aun cuando las imágenes con zoom no funcionaban puesto que al hacer zoom en Aladin no solo aumenta el tamaño, también ajusta las intensidades en los pixels lo cual podría desvirtuar las mediciones y dar

lugar al error. En vistas de que este modelo no tenía arreglo aparente investigue sobre el posible uso de TensorFlow en nuestro proyecto pero las pruebas que hice, empleando tan solo los datos de los pixels, no dieron resultados superiores que los que ya teníamos. Tuve que desechar la idea de emplear esta herramienta de machine learning puesto que la cantidad de imágenes de entrenamiento de que disponemos es muy limitada y las que se requieren para entrenar una red de tensores grande es muy elevada.

En lo referente al detector que empleamos, ideado por mi compañero Daniel, todos colaboramos en la definición de los parámetros que se calcularían de los posibles sistemas y en la implementación de las decisiones referentes a estos, incluidas las cotas en las que han de estar para que se considere el sistema como candidato para albergar una estrella binaria. Además se me ocurrió que, para poner a prueba el nivel de error de nuestras mediciones podría crear un paso que dada una coordenada la busque en el catálogo de WDS y, caso de encontrarla, usando sus datos como referencia calculase cuanto distaban nuestros datos de los originales, fue en este punto cuando implementé la etapa de comprobación.

De la memoria redacté en solitario los capítulos de la Introducción, en español e inglés, el Workflow, el Procesamiento, y traduje el capítulo de Conclusions a inglés. Además colabore con Daniel en la redacción del capítulo de Estrellas dobles, Recolección de datos y la sección 7.2. Como mis compañeros no saben utilizar LaTeX también me he encargado de incluir sus aportaciones a este documento.

11.2. Daniel

Una vez tuvimos el proyecto definitivo, me dediqué al estudio de qué es una estrella doble y cuáles son su características mediante el análisis de decenas de fotografías de estrellas, la lectura de distintos artículos al respecto y la visualización de varios vídeos sobre el tema.

Mis aportaciones a las distintas fases del proyecto han consistido, además de mi participación en todas las reuniones a lo largo del curso, en:

Obtención de imágenes:

- Junto a mis compañeros, creamos dos programas para generar coordenadas y así facilitar la descarga de imágenes, uno con las coordenadas en un rango dado y otro de coordenadas aleatorias. También estudiamos el uso de la herramienta Aladin para la creación de distintos scripts encargados de la descarga de imágenes. Uno para el hemisferio norte y otro para el hemisferio sur.

- Una vez descargadas las imágenes del WDS y varios miles de fotografías sin estrellas dobles, hice un programa para dividirlas en los distintos datasets: training, validation y test, acordes a las necesidades de mis compañeros. Este programa divide las fotografías que sí tienen estrella binaria en los distintos dataset en función de los porcentajes vistos en el apartado 4.4 y añade el mismo número de fotografías sin estrella binaria de manera que los dataset estén equilibrados, descartando las sobrantes.

Diseño de fase de detección:

- Por mi cuenta, estudié distintas técnicas de procesamiento de imágenes y reconocimiento de objetos para encontrar la manera adecuada de reconocer las estrellas en las imágenes.
- Aprovechando que mi compañero David recoloreaba las imágenes de manera que el número de colores se simplificó a rojo, azul, blanco y negro, decidí utilizar la búsqueda de contornos de las estrellas para su análisis.
- Una vez decidida la búsqueda de contornos para encontrar estrellas, investigué distintas bibliotecas dedicadas procesamiento de imágenes hasta encontrar OpenCV, que es la que se utiliza en el proyecto.
- Exploré las distintas posibilidades que ofrece la biblioteca OpenCv y, en base a sus características, diseñé e implementé la aplicación que dedicada al análisis de las características que permiten catalogar a una estrella binaria.

Desarrollo de fase de detección:

- Primero, desarrollé un programa que fuera capaz de encontrar los contornos de las estrellas y sus centros.
- Una vez creado el programa lo amplié de manera que fuera capaz de clasificar las distintas estrellas para poder borrar las que no tuvieran valor a la hora de encontrar estrellas binarias, por ejemplo, estrellas que no se hubieran movido. Esta ampliación fue pensada para poder eliminar el ruido que provocan elementos inútiles en aplicaciones de machine learning como tensorFlow. Esta ampliación fue desechada debido a que el número de imágenes con estrellas dobles es demasiado pequeño como para que pueda aprender correctamente.
- Para aprovechar el primer programa creado, y en vistas de que no íbamos a poder aplicar técnicas de machine learning, decidí mejorar

el primer programa y hacer que calculara los centros antes y después para lo que tuve que añadirle una nueva funcionalidad que permitiera crear dos imágenes que representaran la disposición de las estrellas antes y después. Además, calcula el área de las estrellas de manera que podemos saber cuál es la primaria y cuál la secundaria.

- Una vez tuve los centros correctamente calculados creamos el algoritmo que permite calcular los valores que utiliza el WDS para catalogar las estrellas y comprobamos que eran bastante similares a los obtenidos mediante nuestra aplicación.
- También participé, junto a mis compañeros, en el estudio e implementación de las diferentes comprobaciones que son necesarias para, con los datos calculados, discernir si se trata de una estrella binaria o no.

Memoria:

- Redacción de la sección 3.1 sobre centroides.
- Redacción de la sección 7.1.
- Redacción de las Conclusiones.
- Colaboración en la redacción de la sección 7.2.
- Colaboración en la redacción de la sección 4.4 sobre datasets.
- Modificación de los programas de la etapa de detección, capítulo 7, para guardar las imágenes que se han añadido a la memoria a fin de clarificar el desarrollo de dicha fase.
- Edición de figuras mediante gimp para facilitar la comprensión de conceptos de la fase de análisis del capítulo 7 y conclusiones.

11.3. Javier

A lo largo de la ejecución de este proyecto yo me he encargado de la creación de los modelos de Machine Learning encargados de realizar las predicciones de estrellas dobles y he ayudado encontrar los valores más apropiados de los atributos vistos en el capítulo 7 que se utilizan para ajustar cuándo hay una estrella doble. Así mismo, de la memoria me encargué de redactar el capítulo 8.

En primera instancia colaboré junto a mis compañeros para crear un generador de coordenadas aleatorio y continuo para crear un fichero de coordenadas aleatorias. Con este fichero se descargaron las primeras imágenes,

en ese momento mis compañeros y yo pensamos en los tipos de análisis que se podrían realizar, yo me decanté por realizar un análisis cromático de la imagen y realizar las predicciones sobre si había en una imagen o no un sistema binario de estrellas.

Los objetivos que buscábamos con esta implementación es que cuando el programa global se ejecutara no siguieran adelante la mayor cantidad posible de imágenes que no tuvieran un sistema binario.Para la realización de este analizador cromático nos decantamos por el desarrollo de un modelo de Machine Learning, concretamente de Regresión Logística. Una vez decidido esto, me dispuse a buscar información acerca de Machine Learning y a investigar cómo podría realizar dicho modelo.

Esta investigación me llevó a la biblioteca de Scikit-learn en Python, y mi investigación se centró en la realización de estos modelos en esta biblioteca y de sus posibilidades y limitaciones.

Una vez hube asentado los conocimientos base me dispuse a diseñar un modelo, que a través de la información cromática de una determinada imagen, pudiera determinar si había un sistema binario en esa imagen o no.

El diseño de este programa me llevó a decidir si la información cromática que iba a analizar debía ser absoluta o relativa(en porcentaje). A priori esta decisión podría parecer algo ingenua, pero el formato de los valores de entrada se acaban convirtiendo, en última instancia, en un factor determinante en el comportamiento del modelo. Al final me decanté por utilizar la el formato relativo dado que evita el llamado "overfitting" o sobreentrenamiento, que se produce cuando se entrena un modelo con exceso de información.

Al disponer de los conjuntos de entrada proporcionado por mis compañero el siguiente trabajo consiste en realizar el entrenamiento del modelo. Este entrenamiento consiste en variar cada uno de los 16 parámetros de los que dispone el modelo hasta encontrar los que proporcionan los resultado más satisfactorios.

Este proceso es uno de los más tediosos puesto que no sirve de nada variar los parámetros "dando palos de ciego", es mucho más efectivo investigar qué es y de qué se encarga cada parámetro para saber con mayor certeza cómo debes variar cada parámetro para obtener un mejor resultado. Investigué en la documentación qué función tenían cada uno de los parámetros del modelo cómo se afectaban entre sí y comprobé que importancia tenía en mi modelo cada uno de ellos y cómo interactuaban entre sí cada vez que se producía una variación en alguno de ellos. Después de mucho insistir, conseguí generar un modelo que satisfacía con creces nuestras expectativas.

Como vimos en el apartado 8 este comportamiento que presentaba el programa fue sólo una ilusión, pues existía un error de fondo que hacía que,

a pesar del excelente comportamiento en las predicciones, el programa no se comportaba como debía.

Una vez hubimos detectado y solucionado el problema, volví a realizar el entrenamiento del modelo, ésta vez sin obtener buenos resultados. Pero tras volver a investigar más cada parámetro de forma individual encontré la forma de mejorar el comportamiento final del programa aumentando la precisión a costa de la sensibilidad, y debido a que no cumplía con nuestros objetivos decidimos que desecharíamos esta fase del flujo del programa principal.

Posteriormente, me dediqué a procesar los datos que identifican a las estrellas dobles mediante un modelo de Machine Learning que encontrara los valores más ajustados posibles que hacen característico a un sistema binario de estrellas.

En esta fase, mi trabajo consistió en realizar un trabajo similar al de detección, pero con la particularidad de que los datos calculados para cada estrella en la imagen pasarían a ser introducidos en un modelo basado en árboles de decisión con la idea de aprovechar el árbol resultante para introducir en la fase de detección los valores concretos en la comprobación de si una estrella cumple con lo especificado que debe cumplir una estrella en un sistema binario.

Para realizar esto debía trabajar con un conjunto de imágenes que contuvieran todas ellas estrellas dobles. Además debía identificar a las dos estrellas que formaban parte del sistema binario. Para ello hablé con mis compañeros y llegamos a la conclusión de que si utilizábamos los atributos del ángulo de posición y la separación, teniendo en cuenta un ligero margen de error, y los comparábamos con los mismos valores que nos proporciona wds en su catálogo para la coordenada en la que está situada el sistema binario, podíamos determinar qué dos estrellas formaban parte del sistema binario y añadirles la clase de estrella doble.

Mientras tanto, estaba investigando cómo obtener el árbol de decisión para así poder identificar qué valor deben tener los atributos que identifican a las estrellas binarias del resto de estrellas. Este fue claramente el proceso más largo y complicado de todos, puesto que la información acerca de este atributo era muy escasa.

Desgraciadamente descubrí que el árbol de decisión que se podía obtener no nos daba el valor de los atributos que definen el hiperplano que separan muestras de estrellas que no son dobles de las que sí, en su lugar, proporcionaba un vector de matrices, una por cada estimador del modelo, la información que obtenías para cada muestra es si esta atravesaba un nodo concreto o no. Esta información puede ser útil en muchos casos, pero en el nuestro no, dado que nos proporcionaba la información que precisábamos.

Sin embargo, más tarde descubrí una funcionalidad propia de un clasificador basado en un árbol de decisión que proporcionaba la representación de dicho árbol con toda la información que necesitábamos en cada nodo.

Para poder visualizar este árbol utilicé una biblioteca de visualización de grafos, que con el archivo que había generado en el paso anterior era capaz de generar la imágen de representación del árbol de decisión generado por el modelo.

El inconveniente de esta representación es que dado que no es una estructura de datos lo que se encuentra en archivo, sino una representación de una imagen, si se quiere analizar el árbol no se puede hacer de forma automática y hay que analizarlo a mano.

Independientemente de esto, me dispuse a continuación a generar el modelo, sin embargo, la precisión más alta que conseguí en la clase del sí con este clasificador, no era lo suficientemente alta para utilizar los criterios de decisión obtenidos de los valores de los atributos.

Debido a este problema decidimos descartar esta fase para obtener los criterios de parametrización, con lo que desafortunadamente todo este desarrollo terminó resultando infructuoso al final.

Bibliografía

Bos, W. v. d. Measures of double stars on photographic plates taken by w. j. luyten. 1923. Disponible en `https://openaccess.leidenuniv.nl/handle/1887/5609` (último acceso, Mayo, 2018).

Greaves, J. *New Northern hemisphere common proper-motion pairs*. Monthly Notices of the Royal Astronomical Society, Volume 355, Issue 2, pp. 585-590., 2004. ISBN 2004MNRAS.355..585G.

Hartkopf, W., Harmanec, P. y Guinan, E. *Binary Stars as Critical Tools and Tests in Contemporary Astrophysics (IAU S240)*. Cambridge, 2007. ISBN 9780521863483.

Luyten, W. J. Double stars with common proper motion. 1971. Disponible en `http://adsbit.harvard.edu/full/1971Ap%26SS..11...49L` (último acceso, Mayo, 2018).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. y Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, vol. 12, páginas 2825–2830, 2011.